

**Université Cadi Ayyad
École Supérieure De Technologie-Safi
Département : Informatique
Filière : genie informatique**

Getion des employes

Rapport du TP1 java avance

Réalisé par : ELAJAMI Salma

Encadré par : Mme. KACHBAL Ilham

Année Universitaire : 2024/2025

Table des matières

Inroduction	4
Outils & environnement de travail	5
1 Environnement de travail	5
2 Outils de travail	5
3 Language de Programmation	6
1 Réalisation	7
1 Création de la base de donnée	7
1.1 Script base de donnée	7
2 Architecture MVC (Model-View-Controller)	8
2.1 Implémentation du modèle (couche Model)	8
2.2 Gestion des données (DAO)	11
2.3 Contrôleur (couche Controller)	13
2.4 Interface graphique (couche View).....	15
2.5 Main	20
2 Résultats	21
1 Ajout	21
2 Modification.....	22
3 Suppression	22

Table des figures

1	intellij idea logo	5
2	MySQL Workbench logo	5
3	xampp logo	6
4	java developpement kit logo	6
5	java logo	6
2.1	Interface Utilisateur.....	21
2.2	Affichage de l'Ajout	21
2.3	Affichage de modification.....	22
2.4	Affichage de suppression	22

Inroduction

Ce travail pratique (TP) a pour objectif de développer une application Java dédiée à la gestion des employés en s'appuyant sur l'architecture MVC (Modèle-Vue-Contrôleur) et en utilisant des interfaces graphiques conçues avec la bibliothèque Swing.

Ce projet pédagogique vise à consolider les connaissances en programmation orientée objet (POO) tout en appliquant les principes des bonnes pratiques de conception logicielle. Il a pour but de créer une application intuitive et modulaire, permettant une gestion efficace des données des employés grâce à des fonctionnalités telles que l'ajout, la modification, la suppression et l'affichage des informations.

En proposant une structure organisée et une séparation claire des responsabilités, ce TP constitue une opportunité d'approfondir les concepts de maintenabilité et d'évolutivité, tout en préparant les étudiants à aborder des projets logiciels plus complexes à l'avenir.

Outils & environnement de travail

1 Environnement de travail



FIGURE 1 – intellij idea logo

- **IntelliJ IDEA** : est un IDE développé par JetBrains, conçu principalement pour Java, mais prenant en charge d'autres langages comme Kotlin, Scala et Python. Il se distingue par ses outils intelligents, sa complétion de code avancée, son analyse en temps réel et son intégration avec des outils comme Maven, Gradle et Git, offrant une productivité accrue aux développeurs.

2 Outils de travail



FIGURE 2 – MySQL Workbench logo

- **MySQL Workbench** : est un outil graphique développé pour simplifier la conception, l'administration et la gestion des bases de données MySQL. Il offre une interface intuitive qui permet de manipuler les bases de données visuellement, sans dépendre uniquement des commandes en ligne, rendant les tâches plus accessibles et efficaces.



FIGURE 3 – xampp logo

- **XAMPP** : est une solution open-source qui regroupe Apache, MySQL, PHP, et Perl pour faciliter le développement et le déploiement d'applications web. Il est conçu pour créer un environnement de serveur local rapide et simple, adapté aux développeurs souhaitant tester ou gérer des projets web sur leur machine avant le déploiement.



FIGURE 4 – java developpement kit logo

- **java developpement kit** : est un ensemble d'outils logiciels indispensables pour le développement d'applications Java. Il comprend des composants essentiels tels qu'un compilateur, une machine virtuelle Java (JVM), des bibliothèques standard, et des outils de débogage, permettant de coder, compiler, exécuter et tester des programmes Java efficacement.

3 Language de Programmation



FIGURE 5 – java logo

- **Java** : un langage de programmation orienté objet et une plateforme largement utilisée pour le développement d'applications logicielles. Il a été créé par Sun Microsystems (maintenant propriété d'Oracle) en 1995 et reste l'un des langages les plus populaires au monde, notamment pour les applications d'entreprise, le développement mobile (Android) et les applications web.

Réalisation

1 Création de la base de donnée

1.1 Script base de donnée

```
1 create database DBperson;
2 use DBperson;
3 CREATE TABLE employers (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     first_name VARCHAR(100),
6     last_name VARCHAR(100),
7     email VARCHAR(50),
8     phone INT(10),
9     salary NUMERIC,
10    role VARCHAR(50),
11    poste VARCHAR(50)
12 );
13
14 CREATE TABLE Role (
15     name varchar(30) NOT NULL
16 );
17 INSERT INTO Role (name) VALUES ('Admin'), ('Employee');
18 INSERT INTO Poste (name) VALUES ('Ingenieur'), ('Team_Leader'), ('Pilote');
19
20 CREATE TABLE Poste (
21     name varchar(30) NOT NULL
22 );
```

Listing 1.1 – Script SQL de la base de données

•

2 Architecture MVC (Model-View-Controller)

L'architecture MVC est un modèle de conception qui sépare les responsabilités au sein d'une application, facilitant ainsi la gestion et la maintenance du code. Elle repose sur trois composants principaux :

2.1 Implémentation du modèle (couche Model)

Employer

```
1 package model;
2
3 import enums.*;
4
5 public class Employer {
6
7     private int id;
8     private String firstName;
9     private String lastName;
10    private String email;
11    private int phoneNumber;
12    private double salary;
13    private Role role;
14    private Poste poste;
15
16    public Employer(int id, String firstName, String lastName, String email, int
    phoneNumber, double salary, Role role, Poste poste) {
17        this.id = id;
18        this.firstName = firstName;
19        this.lastName = lastName;
20        this.email = email;
21        this.phoneNumber = phoneNumber;
22        this.salary = salary;
23        this.role = role;
24        this.poste = poste;
25    }
26
27    public int getId() {
28        return id;
29    }
30
31    public String getFirstName() {
32        return firstName;
33    }
34
35    public String getLastName() {
36        return lastName;
37    }
38
39    public String getEmail() {
40        return email;
41    }
42
43    public int getPhoneNumber() {
```



```
44         return phoneNumber;
45     }
46
47     public double getSalary() {
48         return salary;
49     }
50
51     public Role getRole() {
52         return role;
53     }
54
55     public Poste getPoste() {
56         return poste;
57     }
58 }
```

EmployerLogic

```
1 package model;
2
3 import enums.*;
4 import dao.EmployerDAO;
5 import java.util.List;
6
7 public class EmployerLogic {
8
9     private EmployerDAO dao;
10
11     public EmployerLogic (EmployerDAO dao) {
12         this.dao = dao;
13     }
14
15     public boolean addEmployer(int id, String firstName, String lastName, String email,
16 int phoneNumber, double salary, Role role, Poste poste) {
17
18         if ( isValidEmail(email) ) {
19             return dao.addEmployer( new Employer(
20                 id,
21                 firstName,
22                 lastName,
23                 email,
24                 phoneNumber,
25                 salary,
26                 role,
27                 poste
28             ));
29
30         return false;
31     }
32
33
34     public boolean updateEmployer(int id, String firstName, String lastName, String
35 email, int phoneNumber, double salary, Role role, Poste poste) {
```

```
36
37     if ( isValidEmail(email) ) {
38
39         Employer employer = new Employer(
40             id,
41             firstName,
42             lastName,
43             email,
44             phoneNumber,
45             salary,
46             role,
47             poste
48         );
49
50         return dao.updateEmployer(employer);
51     }
52
53     return false;
54 }
55
56 private boolean isValidEmail(String email) {
57     return email.contains("@gmail.com") ? true : false;
58 }
59
60
61 public boolean deleteEmployer(int id) {
62     return dao.deleteEmployer(id);
63 }
64
65 public List<Employer> getAllEmployers() {
66     return dao.getAllEmployers();
67 }
68
69 }
```

2.2 Gestion des données (DAO)

DBConnection

```
1
2
3
4         ;
5         ;
6
7
8
9         .ng URL = "jdbc:mysql://localhost:3306/DBperson";
10        ing USERNAME = "root";
11        ing PASSWORD = "";
12
13        getConnection() throws SQLException {
14        .getConnection(URL, USERNAME, PASSWORD);
15
16
```

EmployerDAO

```
1
2
3
4
5
6
7
8
9
10        .ements EmployerInterface {
11
12        ction;
13
14
15
16        onnection.getConnection();
17        i connectionException) {
18        .on.printStackTrace();
19
20
21
22
23
24        r(Employer employer) {
25        nt addStatement = connection.prepareStatement(
26        employers (first_name, last_name, email, phone,
27        ES (?, ?, ?, ?, ?, ?, ?)) {
28
29        String(1, employer.getFirstName());
30        String(2, employer.getLastName());
31        String(3, employer.getEmail());
```

```
31         addStatement.setInt(4, employer.getPhoneNumber());
32         addStatement.setDouble(5, employer.getSalary());
33         addStatement.setString(6, employer.getRole().name());
34         addStatement.setString(7, employer.getPoste().name());
35
36         return addStatement.executeUpdate() > 0;
37
38     } catch (SQLException addException) {
39         addException.printStackTrace();
40         return false;
41     }
42 }
43
44
45 @Override
46 public boolean updateEmployer(Employer employer) {
47     try (PreparedStatement updateStatement = connection.prepareStatement("
UPDATE employers SET first_name = ?, last_name = ?, email = ?, phone = ?,
salary = ?, role = ?, poste = ? WHERE id = ?")) {
48
49         updateStatement.setString(1, employer.getFirstName());
50         updateStatement.setString(2, employer.getLastName());
51         updateStatement.setString(3, employer.getEmail());
52         updateStatement.setInt(4, employer.getPhoneNumber());
53         updateStatement.setDouble(5, employer.getSalary());
54         updateStatement.setString(6, employer.getRole().name());
55         updateStatement.setString(7, employer.getPoste().name());
56         updateStatement.setInt(8, employer.getId());
57
58         return updateStatement.executeUpdate() > 0;
59
60     } catch (SQLException updateException) {
61         updateException.printStackTrace();
62         return false;
63     }
64 }
65
66 @Override
67 public boolean deleteEmployer(int id) {
68     try (PreparedStatement deleteStatement = connection.prepareStatement("
DELETE FROM employers WHERE id = ?")) {
69
70         deleteStatement.setInt(1, id);
71         return deleteStatement.executeUpdate() > 0;
72
73     } catch (SQLException deleteException) {
74         return false;
75     }
76 }
77
78 @Override
79 public List<Employer> getAllEmployers() {
80     List<Employer> employers = new ArrayList<>();
81     try (ResultSet getResult = connection.prepareStatement("SELECT * FROM
employers").executeQuery()) {
```

```
82
83         while (getResult.next()) {
84             employers.add(new Employer (
85                 getResult.getInt("id"),
86                 getResult.getString("first_name"),
87                 getResult.getString("last_name"),
88                 getResult.getString("email"),
89                 getResult.getInt("phone"),
90                 getResult.getDouble("salary"),
91                 Role.valueOf(getResult.getString("role")),
92                 Poste.valueOf(getResult.getString("poste"))
93             ));
94         }
95
96         } catch (SQLException getException) {
97             getException.printStackTrace();
98         }
99         return employers;
100     }
101 }
```

2.3 Contrôleur (couche Controller)

EmployerController

```
1 package controller;
2
3 import view.*;
4 import dao.*;
5 import model.*;
6 import enums.*;
7 import java.util.List;
8 import javax.swing.JOptionPane;
9
10 public class EmployerController {
11
12     private FormFrame frame;
13     private EmployerLogic employerLogic;
14
15     public EmployerController(FormFrame frame, EmployerLogic employerLogic) {
16         this.frame = frame;
17         this.employerLogic = employerLogic;
18
19         frame.getBtnPanel().getAddBtn().addActionListener(addEvent -> addEmployer
20             ());
21         frame.getBtnPanel().getUpdateBtn().addActionListener(updateEvent ->
22             updateEmployer());
23         frame.getBtnPanel().getRemoveBtn().addActionListener(deleteEvent ->
24             deleteEmployer());
25         loadEmployers();
26     }
27
28     private void addEmployer() {
```

```

26         try {
27
28             if (employerLogic.addEmployer(
29                 1,
30                 frame.getInPanel().getFirstNameField().getText(),
31                 frame.getInPanel().getLastNameField().getText(),
32                 frame.getInPanel().getEmailField().getText(),
33                 Integer.parseInt(frame.getInPanel().getTelephoneField().
getText()),
34                 Double.parseDouble(frame.getInPanel().getSalaryField().getText
35                 ()),
36                 Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
37                 Poste.valueOf(frame.getInPanel().getSelectedPoste().toString())
38             ))
39             {
40                 JOptionPane.showMessageDialog(frame, "Employer added successfully
.");
41                 loadEmployers();
42             } else {
43                 JOptionPane.showMessageDialog(frame, "Failed to add employer.");
44             }
45         } catch (Exception e) {
46             JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage
47             ());
48         }
49
50         private void updateEmployer() {
51             try {
52                 if (employerLogic.updateEmployer(
53                     frame.getListPanel().getSelectedRowId(),
54                     frame.getInPanel().getFirstNameField().getText(),
55                     frame.getInPanel().getLastNameField().getText(),
56                     frame.getInPanel().getEmailField().getText(),
57                     Integer.parseInt(frame.getInPanel().getTelephoneField().
getText()),
58                     Double.parseDouble(frame.getInPanel().getSalaryField().getText()
59                     ),
60                     Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
61                     Poste.valueOf(frame.getInPanel().getSelectedPoste().toString())
62                 ))
63                 {
64                     JOptionPane.showMessageDialog(frame, "Employer updated successfully
.");
65                     loadEmployers();
66                 } else {
67                     JOptionPane.showMessageDialog(frame, "Failed to update employer.");
68                 }
69             } catch (Exception e) {
70                 JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
71             }
72
73             private void deleteEmployer() {

```

```
73     try {
74         if (employerLogic.deleteEmployer(frame.getListPanel().getSelectedRowId())
75     ) {
76             JOptionPane.showMessageDialog(frame, "Employer deleted successfully
77             .");
78             loadEmployers();
79         } else {
80             JOptionPane.showMessageDialog(frame, "Failed to delete employer.");
81         }
82     } catch (Exception e) {
83         JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
84     }
85     }
86
87     private void loadEmployers() {
88         frame.getListPanel().updateEmployerList(employerLogic.getAllEmployers());
89     }
90 }
```

2.4 Interface graphique (couche View)

BtnPanel

```
1 package view;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class BtnPanel extends JPanel {
7
8     private JButton addBtn, removeBtn, updateBtn;
9
10    public BtnPanel() {
11        setLayout(new FlowLayout());
12
13        addBtn = new JButton("Add");
14        removeBtn = new JButton("Remove");
15        updateBtn = new JButton("Update");
16
17        add(addBtn);
18        add(removeBtn);
19        add(updateBtn);
20    }
21
22    public JButton getAddBtn() {
23        return addBtn;
24    }
25
26    public JButton getRemoveBtn() {
27        return removeBtn;
28    }
29
30    public JButton getUpdateBtn() {
```

```
31         return updateBtn;
32     }
33 }
```

FormFrame

```
1
2 package view;
3
4 import javax.swing.*;
5 import java.awt.*;
6
7 public class FormFrame extends JFrame {
8
9     private InputPanel inPanel;
10    private BtnPanel btnPanel;
11    private ListPanel listPanel;
12
13    public FormFrame() {
14        setTitle("Person Form");
15        setSize(800, 600);
16        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17        setLayout(new BorderLayout());
18
19        inPanel = new InputPanel();
20        btnPanel = new BtnPanel();
21        listPanel = new ListPanel();
22
23        add(inPanel, BorderLayout.NORTH);
24        add(btnPanel, BorderLayout.SOUTH);
25
26        JScrollPane scrollPane = new JScrollPane(listPanel);
27        add(scrollPane, BorderLayout.CENTER);
28
29        setVisible(true);
30    }
31
32    public InputPanel getInPanel() {
33        return inPanel;
34    }
35
36    public BtnPanel getBtnPanel() {
37        return btnPanel;
38    }
39
40    public ListPanel getListPanel() {
41        return listPanel;
42    }
43 }
```

InputPanel

```
1 package view;
```



```
2
3 import javax.swing.*;
4 import java.awt.*;
5 import enums.Role;
6 import enums.Poste;
7
8 public class InputPanel extends JPanel {
9
10     JTextField firstNameField, lastNameField, emailField, telephoneNumberField,
    salaryField;
11     JComboBox<Role> roleField;
12     JComboBox<Poste> posteField;
13
14     public InputPanel() {
15         setLayout(new GridLayout(7, 2, 5, 5));
16         setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
17
18         firstNameField = new JTextField(15);
19         lastNameField = new JTextField(15);
20         emailField = new JTextField(15);
21         telephoneNumberField = new JTextField(15);
22         salaryField = new JTextField(15);
23         roleField = new JComboBox<>(Role.values());
24         posteField = new JComboBox<>(Poste.values());
25
26         add(new JLabel("First Name"));
27         add(firstNameField);
28
29         add(new JLabel("Last Name"));
30         add(lastNameField);
31
32         add(new JLabel("Email"));
33         add(emailField);
34
35         add(new JLabel("Telephone Number"));
36         add(telephoneNumberField);
37
38         add(new JLabel("Salary"));
39         add(salaryField);
40
41         add(new JLabel("Role"));
42         add(roleField);
43
44         add(new JLabel("Poste"));
45         add(posteField);
46     }
47
48     public JTextField getFirstNameField() {
49         return firstNameField;
50     }
51
52     public JTextField getLastNameField() {
53         return lastNameField;
54     }
55 }
```

```
56     public JTextField getEmailField() {
57         return emailField;
58     }
59
60     public JTextField getTelephoneNumberField() {
61         return telephoneNumberField;
62     }
63
64     public JTextField getSalaryField() {
65         return salaryField;
66     }
67
68     public Role getSelectedRole() {
69         return (Role) roleField.getSelectedItem();
70     }
71
72     public Poste getSelectedPoste() {
73         return (Poste) posteField.getSelectedItem();
74     }
75 }
```

ListPanel

```
1 package view;
2
3 import javax.swing.*;
4 import javax.swing.border.LineBorder;
5 import java.awt.*;
6 import java.util.ArrayList;
7 import java.util.List;
8 import model.Employer;
9 import java.awt.event.MouseAdapter;
10 import java.awt.event.MouseEvent;
11
12 public class ListPanel extends JPanel {
13
14     private JPanel contentPanel;
15     private int selectedRowId = -1;
16     private List<JPanel> rowPanels = new ArrayList<>();
17
18     public ListPanel() {
19         setLayout(new BorderLayout());
20
21         JPanel titlePanel = new JPanel();
22         titlePanel.setLayout(new GridLayout(1, 5));
23         titlePanel.add(new JLabel("Id", SwingConstants.CENTER));
24         titlePanel.add(new JLabel("Nom", SwingConstants.CENTER));
25         titlePanel.add(new JLabel("Prenom", SwingConstants.CENTER));
26         titlePanel.add(new JLabel("Email", SwingConstants.CENTER));
27         titlePanel.add(new JLabel("Salaire", SwingConstants.CENTER));
28         titlePanel.setBorder(new LineBorder(Color.BLACK));
29         add(titlePanel, BorderLayout.NORTH);
30
31         contentPanel = new JPanel();
```

```

32     contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.Y_AXIS));
33     JScrollPane scrollPane = new JScrollPane(contentPanel);
34     add(scrollPane, BorderLayout.CENTER);
35 }
36
37 public void updateEmployerList(List<Employer> employers) {
38     contentPanel.removeAll();
39     rowPanels.clear();
40
41     for (Employer employer : employers) {
42         JPanel rowPanel = new JPanel(new GridLayout(1, 5));
43         rowPanel.setBorder(new LineBorder(Color.GRAY));
44         rowPanel.setMaximumSize(new Dimension(Integer.MAX_VALUE, 30));
45
46         JLabel idLabel = new JLabel(String.valueOf(employer.getId()),
47 SwingConstants.CENTER);
48         JLabel lastNameLabel = new JLabel(employer.getLastName(),
49 SwingConstants.CENTER);
50         JLabel firstNameLabel = new JLabel(employer.getFirstName(),
51 SwingConstants.CENTER);
52         JLabel emailLabel = new JLabel(employer.getEmail(), SwingConstants.
53 CENTER);
54         JLabel salaryLabel = new JLabel(String.valueOf(employer.getSalary()),
55 SwingConstants.CENTER);
56
57         rowPanel.addMouseListener(new MouseAdapter() {
58             @Override
59             public void mouseClicked(MouseEvent e) {
60                 highlightRow(rowPanel, employer.getId());
61             }
62         });
63
64         rowPanel.add(idLabel);
65         rowPanel.add(lastNameLabel);
66         rowPanel.add(firstNameLabel);
67         rowPanel.add(emailLabel);
68         rowPanel.add(salaryLabel);
69
70         contentPanel.add(rowPanel);
71         rowPanels.add(rowPanel);
72     }
73
74     contentPanel.revalidate();
75     contentPanel.repaint();
76 }
77
78 private void highlightRow(JPanel selectedRow, int employerId) {
79     for (JPanel row : rowPanels) {
80         row.setBackground(Color.WHITE);
81     }
82
83     selectedRow.setBackground(Color.LIGHT_GRAY);
84     selectedRowId = employerId;
85 }

```

```
82     public int getSelectedRowId() {  
83         return selectedRowId;  
84     }  
85 }
```

2.5 Main

```
1 import view.*;  
2 import dao.*;  
3 import model.*;  
4 import controller.EmployerController;  
5  
6 public class Main {  
7     public static void main(String[] args) {  
8         new EmployerController( new FormFrame(), new EmployerLogic(new EmployerDAO  
9             ())) ;  
10    }
```

Résultats

Person Form

First Name

Last Name

Email

Telephone Number

Salary

Role

Poste

MANAGER

DEVELOPEUR

Id	Nom	Prenom	Email	Salaire
----	-----	--------	-------	---------

Add

Remove

Update

FIGURE 2.1 – Interface Utilisateur

1 Ajout

Person Form

First Name

Last Name

Email

Telephone Number

Salary

Role

Poste

Kamal

Alami

alarni@gmail.com

0078985434

12000

EMPLOYER

CHEF DE PROJET

Id	Nom	Prenom	Email	Salaire
2	elajani		@gmail.com	1200.0
3	elajani		@gmail.com	20000.0

Add

Remove

Update

Message

Employer added successfully.

OK

FIGURE 2.2 – Affichage de l’Ajout

2 Modification

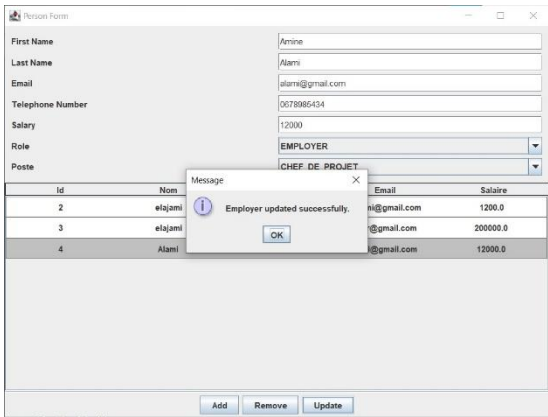


FIGURE 2.3 – Affichage de modification

3 Suppression

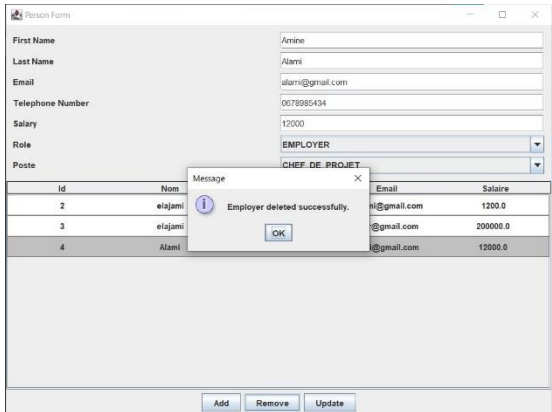


FIGURE 2.4 – Affichage de suppression

Conclusion

En conclusion, ce projet de gestion des employés a permis de mettre en pratique les concepts fondamentaux de la programmation orientée objet ainsi que les principes de l'architecture MVC. À travers le développement d'une application intuitive et modulaire, nous avons pu créer un outil efficace pour la gestion des données des employés, offrant des fonctionnalités de base telles que l'ajout, la modification, la suppression et l'affichage des informations.

Ce travail a également permis d'approfondir des notions essentielles telles que la maintenabilité et l'évolutivité du code, tout en appliquant des bonnes pratiques de conception logicielle. Il s'inscrit comme une première étape dans la maîtrise de l'ingénierie logicielle et prépare ainsi à des projets plus complexes à l'avenir. En somme, cette expérience a renforcé ma compréhension des défis et des exigences liés au développement d'applications robustes et évolutives, tout en mettant en évidence l'importance de la structure et de la gestion efficace du code pour des projets futurs.